(12) UK Patent Application (19) GB (11) 2 312 767 (13) A

(43) Date of A Publication 05.11.1997

(21) Application No 9708608.6

(22) Date of Filing 28.04.1997

(30) Priority Data
(31) 08638807 (32) 29.04.1996 (33) US

(71) Applicant(s)
Mitel Corporation

(Incorporated in Canada - Ontario)

P O Box 13089, Kanata, Ontario K2K 1X3, Canada

(72) Inventor(s)
Richard Deadman

(74) Agent and/or Address for Service
John Orchard & Co
Staple Inn Buildings North, High Holborn, LONDON,
WC1V 7PZ, United Kingdom

(51) INT CL[6]
G06F 1/00

(52) UK CL (Edition O )
G4A AAP

(56) Documents Cited
EP 0570123 A1
Dialog record 01887837 of Computer Shopper, v16, n2, Feb 199 6, p 581(2) Dialog record 01939746 of PC Magazine, v15, n11, 11 June 199 6, p 221(5)

(58) Field of Search
UK CL (Edition O ) G4A AAP
INT CL[6] G06F 1/00
Online: COMPUTER, INSPEC, WPI

# (54) Protected persistent storage access by applets

(57) A method of processing an applet comprises storing a file in a persistent storage medium (PSM), the file including an access control list 20; transmitting an applet from a server, the applet including at least one of applet identification data and a private key encrypted domain identifier and public key pair, a maximum file size indicator, and a specification of required operations; receiving the applet by an application engine 16; in the event the domain identifier is included in the applet, decrypting the domain identifier using the public key; checking at least one of the applet identification data and decrypted domain identifier against the access control list for a match, and in the event a match is found, allowing the operations specified in the applet on a file stored in a persistent storage medium and as specified in the file access control list.

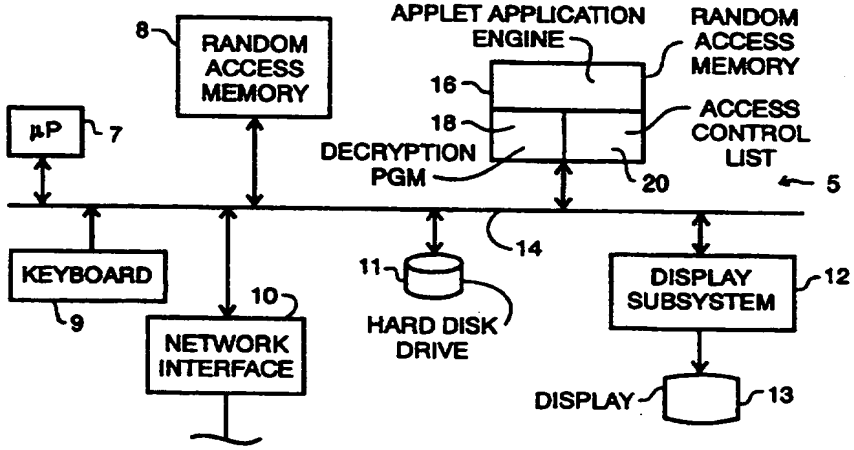


FIG. 2

GB 2 312 767 A

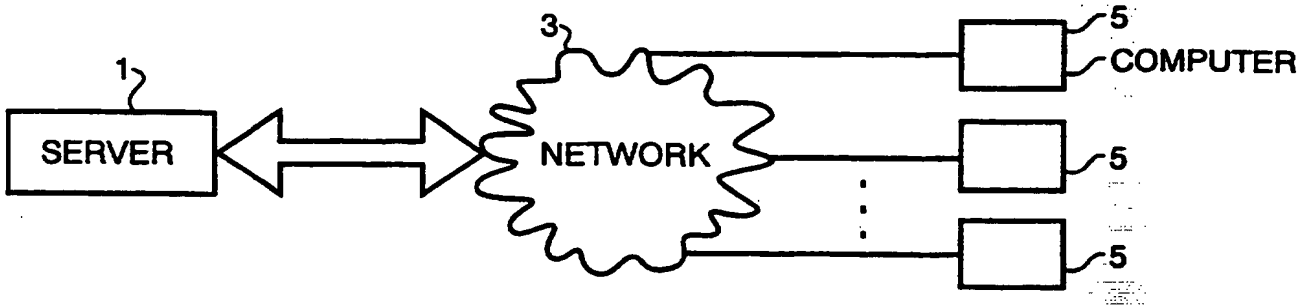


FIG. 1

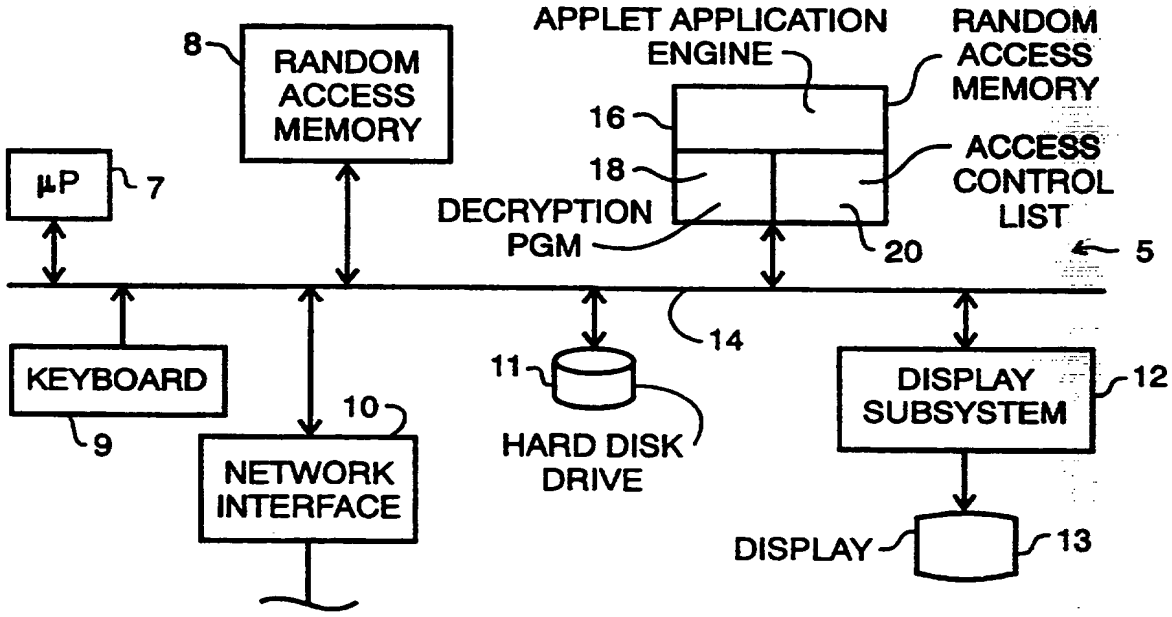


FIG. 2

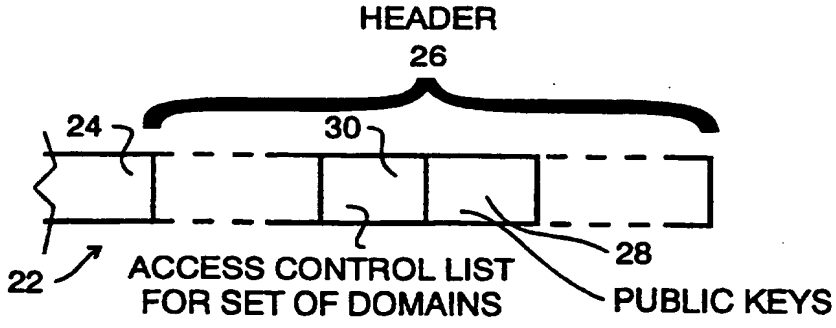


FIG. 3

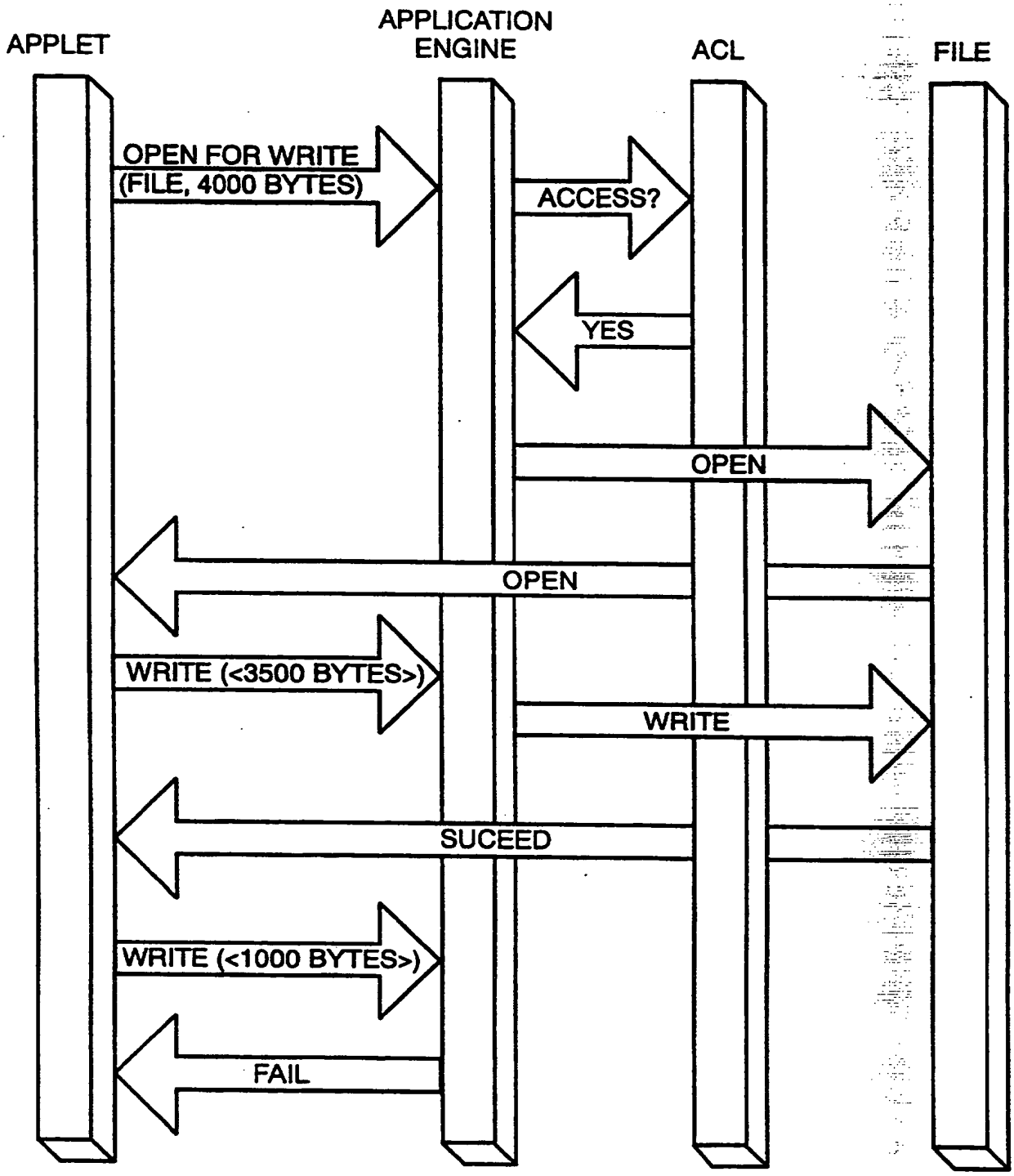
APPLET
APPLICATION ENGINE
ACL
FILE
OPEN FOR WRITE (FILE, 4000 BYTES)
ACCESS?
YES
OPEN
OPEN
WRITE (<3500 BYTES>)
WRITE
SUCEED
WRITE (<1000 BYTES>)
FAIL

FIG. 4

2312767

# PROTECTED PERSISTENT STORAGE ACCESS FOR MOBILE APPLICATIONS

This invention relates to the field of computer networks and in particular to a method of processing computer applets in a secure manner.

With the advent of object based programming, it has become possible to transmit applets via a network such as the internet to various computers connected to the network. An applet is a small self-contained application program which can cause a computer to perform a particular function, and contains both instructions for the computer to operate and data which is to be processed in its operation.

Various programming languages have been designed to create applets, such as Telescript, SafeTCL, Java, CyberAgents and ClearLake Agents. By the use of interpreters, the various applets can be hardware independent, that is, the same applet can be received and processed by computer hardware having different operating systems. For example, the same applet would be able to be received and processed by a MacIntosh operating system and by one of the Microsoft Windows operating systems on an Intel processor based (e.g. IBM) personal computer type operating system.

It has been speculated that the programs and data that computers that receive applets use would provide all that a computer would need, to be able to be fully functional and which would result in a significantly less expensive computer than at present. This is because the computer would be expected to use the network as a mass storage repository of programs and data. Programs would be paid for on a use basis, rather than on an up-front-payment and unlimited use basis. However this presupposes that a user would be prepared to allow mass storage to be out of his control to store

critical data, specially written programs, etc. For the reason of security, it had been believed to be desirable and/or necessary to provide a local mass storage associated with each or with most computers.

With the storage of data on a local mass storage device, the problem of security arises in relation to the reception from the network of applets, since those applets could access and transmit or corrupt files.

The security problem has been handled in one of two ways.

The first is to restrict the use of applets to a closed or managed system, wherein software components can be given domain names and all possible execution sites can be populated with lists of allowed domains. For example, in the CyberAgents program, domains are used to control execution and security on remote sites. A domain is a jurisdiction group of applets that share access rights.

The second way of handling security is to isolate the applets from important system resources, as noted earlier above. Applets received from an uncontrolled system network such as the internet have been restricted from being able to access mass storage devices, and have been relegated to providing fancy graphics and to providing a graphical user interface for client-server computing. Programs to create applets in open uncontrolled systems, are e.g. Telescript, SafeTCL and Java. For example, Java can cause a computer to display a window and provide the content of the window and play sounds, but cannot write to the computer disk drive.

While use of either of these schemes produces the required security to ensure that rogue applets do not cause damage to a computer's resources, such as the

files, they also place restriction on the usability of open system mobile applets. If a downloaded applet cannot access the disk drive to provide persistent local storage for the user, the usability of applets written using such an isolation scheme is very limited.

Operating systems approach security by providing usage rights over directories and files. By assigning the applet application engine to a user group, the applets executed by the application engine are restricted to accessing those files available to the application engine itself. While this provides security to other file system files outside the reach of the application engine, it does not provide security between applets. Similarly, some applet application engines, such as Sun's HotJava, provide access control lists. These lists specify which subset of the application engines accessible files can be accessed by applets. This provides extended security, but does not solve the problem of providing security to data between applets.

HotJava also specifies a method by which access to files can be granted to applets through the use of a dialog box to the user of the applet, wherein the user is required to enter a security authorization. This provides greater security between applets but is intrusive and requires that the user understand the sensitivity and origin (virtual ownership) of data in each file.

If the application engine can determine the source of the applet, access rights can be set according to certain rules. Java can detect if an applet originates within a firewall (a software barrier to outside access to a local (internal) network); the HotJava Browser allows different security rights to applets loaded on each side of the firewall. Read and

write access can be granted to internal applications only.

General Magic's Telescript technology provides a generic set of authorities and permits. Agents within a region all contain the same authority - this is similar to the concept of user domains. Each Telescript agent has only one authority. Permits control the execution of instructions or the access and usage of a resource. Telescript does not specify how a Telescript place negotiates usage rights for a resource with an agent of a certain authority, or stores the access rights for each resource instance.

The term "persistent storage" will be used in this specification as a generic term for all media which store or carry files, data or programs which are to be protected, and can include, but is not restricted to electronic random access memory, floppy or hard disk drives, buses, etc.

Features of an arrangement to be described, by way of example in illustration of the present invention is that it provides comparatively secure access to persistent storage for distributed applet code within an open uncontrolled computing environment. It protects files from unauthorized use by unknown mobile applets while still providing access to the file system for these applets. This involves negotiating access rights for the unknown applet.

In a particular embodiment illustrative of the invention, a method of processing an applet includes storing a file in a persistent storage medium (PSM), the file including an access control list, and transmitting an applet from a server, the applet including at least one of applet identification data and a private key encrypted domain identifier and public key pair, a maximum file size indicator, and a specification of required storage operations, receiving the applet by

an application engine, and in the event the domain identifier is included in the applet, decrypting the domain identifier using the public key, checking at least one of the applet identification data and decrypted domain identifier against the access control list for a match, and in the event a match is found, allowing the operations specified in the applet on a file stored in a persistent storage medium which allow access to the applet as specified in the file access control list.

In another illustrative embodiment, a method of providing an applet includes receiving applets in an application engine, the applets comprising specifications of operations to be performed, and some of the applets being comprised of at least one of applet identification data and private key encrypted domains, processing the applets to an extent to which access to files contained in a persistent storage medium is not required, in the event applets are comprised of private key encrypted domains, decrypting the domains, checking at least one of the applet identification data and decrypted domains against a control list, and processing the applets which the at least one of applet identification data and decrypted domains match entries in the control list to access the files contained in persistent storage medium required by the applets.

In yet another illustrative embodiment, a method of processing an applet includes storing an access control list which contains an identity of applets or domains that can have access to files stored in a persistent medium, and giving access to the access control medium by applets which contain an identity or a domain corresponding to an entry on the list.

In a further illustrative embodiment, an applet application engine includes an access

control list, the engine and list stored in a memory of a computer, and apparatus for providing an applet comprising an identification of at least one of an applet identifier and a domain for comparison with the list and determination of a match, and apparatus for allowing access to an otherwise protected computer resource in the event of the match.

Arrangements which are given by way of example, and which are helpful in understanding the invention, will now be described with reference to the accompanying drawings, in which:

Figure 1 is a block schematic diagram of a network on which the present invention can be implemented,

Figure 2 is a block schematic diagram of a computer used in the network of Figure 1, on which the present invention can be implemented,

Figure 3 is an illustration of a representative applet which may be used in carrying out the present invention, and

Figure 4 illustrates a representative method of operation illustrative of the invention.

Figure 1 illustrates an open and uncontrolled network, and is comprised of a server 1, which interfaces a network 3, to which various computers 5 are or can be connected. The network 3 can be a wide area network, the internet, a telephone network, etc.

In operation, the server 1 applies applets to the network 3 for reception by one or more computers 5. A computer which receives the applet, which is a complete object based program, processes it using the data contained therein. As noted above, in prior art open systems the applet was restricted from being able to access persistent storage.

Figure 2 illustrates the basic architecture of a representative computer system 5. A microprocessor 7, random access memory 8 (RAM), a keyboard 9, a network interface device 10 such as a modem, a hard disk drive 11 persistent storage device, and a display subsystem 12 to which a display 13 is connected, are connected to, or are in communication with, a bus 14.

In addition, random access memory 16 (RAM), which may be part of or merged with RAM 8, is connected or in communication with bus 14. RAM 16 contains a applet application engine, which is comprised of an interpreter for the particular applet language used, and a software command structure to cause microprocessor 7 to process the applets. For example, for the Java language, an applet application engine can be HotJava, available from Sun Microsystems Inc..

In operation of a prior art system, applets are received from the server 1 via the network 3 and network interface 10, are interpreted by microprocessor 7 using the interpreter in application engine 16 and the interpreted instructions are processed by microprocessor 7 to cause the display subsystem 12 to display graphics, etc. on display 13. The application engine will not access persistent storage media such as the hard disk drive 11, to avoid corruption etc. of files as described therein, and therefore cannot use data or other programs stored therein. Such a system is thus considerably limited.

In an arrangement illustrative of the present invention, the application engine 16 contains a public key decryption program 18 and an access control list 20 of authorized applets and/or domains which may be allowed to access persistent storage media.

In the illustrative arrangement, when applets are received via the network interface, access

rights to the persistent storage media are negotiated using the application engine 16. Access rights need not be restricted and access negotiated only to persistent storage media; any resource of the computer or controlled or accessed by the computer can be restricted and access thereto negotiated.

Access rights are negotiated by

(a) authentication; the applet application engine verifies the identity or domain of all applets requesting access to persistent storage;

(b) usage rights negotiation; this involves verifying the applet's authority to read, write or append to a file. Methods to ensure that an applet has access rights to a particular file are described below. It should be noted that in an open uncontrolled system, central authentication or domain management is not possible. For this reason, the negotiation of access rights is conducted between the application engine, the file and the applet;

(c) quality of service; maximum disk usage for an application is set. Limits are preferably placed on access rights which give the bounds within which the granted resource may be used.

By acting as an intermediary between the unknown applet and the disk resource, the application engine negotiates and enforces access rights. For disk files, this involves acting as an intermediary for all disk actions. If a target file will grow beyond a predetermined size, a write can be denied or the user can be asked for override permission, which effectively allows growth of the access rights by an incremented amount. The application engine can specify an override zone, above which the access right will not be allowed to grow.

As a safety mechanism, file versioning may be provided by the application engine to ensure that any modifications or deletions to files can be recovered in the event of damage to the file system. Access control list 20 is used by the application engine to specify what default and file specific access rights are for individual persistent storage objects (files).

Each applet should be associated with a server site that uniquely identifies the applet. If loaded from the World Wide Web, for example, this identity can be the universal resource locator (URL) of the applet. When an applet creates a new file, it can specify which applets have which rights on the file.

Domains are groups of applets which share access rights. Using private key encryption and an encrypted message supplied by the applet, the application engine can ensure that an applet belongs to a specified domain. Applet files contain a header which contains the access control list for a set of domains and public keys for each domain.

Figure 3 illustrates an applet 22, which contains a payload 24 comprised of an object based application program including data, and a header 26. The header 26 contains a public key or keys 28, and an application control list 30. It also contains an identification of the applet (e.g. the URL if used on the World Wide Web), which can be encrypted at the server using a private key, for decryption under control of the application engine 16 using a public key 28. The header also contains an encrypted set of domains e.g. domain messages, which can be decrypted using a public key 28.

An applet can belong to more than one domain at a time. It is the responsibility of the domain

owners to ensure that the private key used to generate encrypted domain authentication messages is kept secure.

As noted above, the encrypted message may encrypt the applet identification using the private key. This ensures that the encrypted message has not been copied from another applet in an attempt to borrow that applet's domain capabilities. Any attempt to clone the server applet to another machine and then be able to read domain owned files would then cause the encrypted message to not match the applet identification, invalidating the applet's domain within the application engine.

A combination of access methods is preferably used to provide secure persistent storage access to applets. Both applet identification and preferably domains are implemented as keys within the access control list for a file.

The language classes for file access in the application engine 16 for the particular language used, e.g. Java should be amended to allow access to the file system based on the applet's identification and domain. Applets should be modified as indicated above with respect to Figure 3 to optionally include their domain, and by the use of public key encryption thereof and/or their identification. Each file with access rights associated with it is contained in the access control list 20 (Figure 2) in a file stored in RAM 16 accessible by the application engine, e.g. a browser program. The application engine retained in RAM 16 negotiates access rights for each loaded applet. Access rights contain quality of service information that defines how each resource can be accessed, including maximum size changes. Each change to the file system is checked to see if it would violate access rights. This includes

the right to access, change or create a file and limits on the size of changes.

The structure of the access control list is preferred to be:

<Filename:...>
<<AccessMode,...>:<IDorDomain,...>;...>

where:

| | |
|---|---|
| Filename | Contains one or more file names, each of which may contain wild-cards |
| AccessMode | Is one of Read, Write, Append, Delete, NewFile, List, Query, (MMaxSize:x) |
| IDorDomain | Is either a URL specifying an applet, or a (DomainName:publicKey) pair |

For example, an entry where all files in the "/public" directory are readable, "/private" can have new files created by applet, URL "http://www.foo.bar/applet.class" and applets of domain "MyDomain" have write access to the file "/private/foo.bar" with add mode set to 5000 bytes, could appear as:

/public/*
Read : *
/private/*
NewFile : *
/private/foo.bar
Write, (MaxSize:5000) :
http://www.foo.bar/applet.class; (MyDomain: 987654321)

A message send within the application engine would appear as shown in the flow chart of Figure 4, in which time runs from top to bottom of the figure.

In this example the applet requests to open the file to write 4000 bytes. The access control list (ACL) is queried by the application engine. The request

is approved, and the file is opened. When 3500 bytes are added, the write to the file succeeds. An attempt to write a further 1000 bytes does not succeed and they are not written, since they exceed the application engine's stored access rights (4000 bytes) for the applet over the file.

Thus it may be seen that there has been described a method by which applets can access files stored on persistent storage media, or can access other computer system resources in an open system that would otherwise be protected, in a secure manner, without concern that a rogue applet has adopted the domain of another applet, or concern that a file can be stolen or corrupted. The method also provides protection against overrunning the storage media due to excessive file size production. Yet at the same time it allows more sophisticated functioning of a computer connected to an open system, since files and other resources at, controlled by or connected to the local computer can be used in processing of applets.

A person understanding this description may now conceive of alternative structures and embodiments or variations of the above. All those which fall within the scope of the claims appended hereto are considered to be part of the present invention.

## CLAIMS

1. A method of processing an applet including

(a) storing a file in a persistent storage medium (PSM), the file including an access control list,

(b) transmitting an applet from a server, the applet including at least one of applet identification data and a private key encrypted domain identifier and public key pair, a maximum file size indicator, and a specification of required operations,

(c) receiving said applet by an application engine,

(d) in the event the domain identifier is included in the applet, decrypting the domain identifier using the public key,

(e) checking said at least one of said applet identification data and decrypted domain identifier against the access control list for a match, and

(f) in the event a match is found, allowing said operations specified in the applet on a file stored in a persistent storage medium which allow access to the applet as specified in the file access control list.

2. A method as defined in claim 1 including terminating performing of said operation specified in the applet in the event said operation would result in a file size in the persistent storage medium that exceeds said maximum file size indicated in the applet.

3. A method as defined in claim 1 including not performing said operation in the event a match is not found.

4. A method of processing applets including

(a) receiving applets in an application engine, said applets comprising specifications of operations to be performed, and some of said applets being comprised of at least one of applet identification data and private key encrypted domains,

(b) processing said applets to an extent to which access to files contained in a persistent storage medium is not required,

(c) in the event applets are comprised of private key encrypted domains, decrypting said domains,

(d) checking said at least one of said applet identification data and decrypted domains against a control list, and

(e) processing those of said applets of which said at least one of applet identification data and decrypted domains match entries in said control list to access said files contained in persistent storage medium required by said applets.

5. A method of processing an applet including storing an access control list which contains an identity of applets or domains that can have access to files stored in a persistent medium, and giving access to said persistent medium by applets which contain an identity or a domain corresponding to an entry on the list.

6. A method as defined in claim 5 including using an application control engine stored in a computer to inspect said applets, check said access control list and process said applets identified on said access control list to operate on a file stored in said access control medium.

7. A method as defined in claim 6 including checking an applet for a file size limitation specification carried by said applet, and failing to operate on said file in the event operation specified by the applet would result in a file size in excess of said file size.

8. A method as defined in claim 5 in which at least one of said identity and domain carried by the applet is private key encrypted and includes a corresponding public key, and including the steps of decrypting to authenticate said at least one of said identity and domain using the public key, said giving access step being carried out only in the event the decrypted said at least one of said identity and domain is authenticated.

9. An applet application engine including an access control list, said engine and list being stored in a memory of a computer, and means for providing an applet having an identification of at least one of an applet identifier and a domain for comparison with said list and determination of a match, and means for allowing access to an otherwise protected computer resource in the event of said match.

10. An engine as claimed in claim 9, in which the computer resource is a persistent storage medium.

11. An engine as claimed in claim 9 further including a public key decryption means, said applet having a private key encrypted identifier or domain and public key, for decryption by said decryption means.

12. A computer software applet including a private key encrypted authentication code defining a domain of said applet.

13. A method as claimed in any one of claims 1, 4, or 5 substantially as described herein with reference to the accompanying drawings.

14. An applet application engine as claimed in claim 9 substantially as described herein with reference to the accompanying drawings.

15. A computer software applet as claimed in claim 12 substantially as described herein with reference to the accompanying drawings.

The Patent Office

17

**Application No:** GB 9708608.6 **Examiner:** B G Western
**Claims searched:** 1-11, 13, 14 **Date of search:** 15 July 1997

# Patents Act 1977
# Search Report under Section 17

**Databases searched:**

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A AAP

Int Cl (Ed.6): G06F 1/00

Other: On-line databases: COMPUTER, INSPEC, WPI

**Documents considered to be relevant:**

| Category | Identity of document and relevant passage | Relevant to claims |
|---|---|---|
| X | EP-0570123-A1 (FISCHER) N.b. pages 5-13 | 1,3-6,9,10 |
| X | Dialog record 01887837 of Computer Shopper, v16, n2, Feb 1996, p581(2), Nicolaisen N, "Waking up the Web: Sun Microsystem's Java gives programming on the Web a boost" | 1,3-6,9,10 |
| X | Dialog record 01939746 of PC Magazine, v15, n11, June 11 1996, p221(5), Lam J, "Java and Java tools: waking up the Web" | 1,3-6,9,10 |

| | | | |
|---|---|---|---|
| X | Document indicating lack of novelty or inventive step | A | Document indicating technological background and/or state of the art. |
| Y | Document indicating lack of inventive step if combined with one or more other documents of same category. | P | Document published on or after the declared priority date but before the filing date of this invention. |
| & | Member of the same patent family | E | Patent document published on or after, but with priority date earlier than, the filing date of this application. |